Sponsored by

Advanced Research Projects Agency

ARPA Order 2311

ARPA Program Code No.3D20

Contractor:

San Diego State University Foundation
5402 College Avenue
San Diego, California 92115

Title:Advanced Computer-Based Instruction

Effective Date of Contract: June 19, 1973

Contract Expiration Date: June 30, 1974

Amount of Contract: $ 126,702.85

Contract No. N61339-73-C-0184

Principal Investigators and Project Scientists:

Frederick Wm.Hornbeck,Ph.D. and Joseph R.Levine,Ph.D.

Department of Psychology
San Diego State University
5402 College Avenue
San Diego, California 92115

Telephone:(714)286-5823 or 286-5358

AD-774 929

ADVANCED COMPUTER-BASED INSTRUCTION--PROJECT CAMELOT: PRELIMINARY REPORT ON A CAI SYSTEM

Lynn H. Brock, et al

California State University

Prepared for:

Naval Training Equipment Center
Advanced Research Projects Agency

21 February 1974

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-774929

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Project CAMELOT: Preliminary Report on a CAI System	5. TYPE OF REPORT & PERIOD COVERED: Quarterly/ October 1 - December 31, 1973
	6. PERFORMING ORG. REPORT NUMB:
7. AUTHOR(s): Lynn H. Brock/Frederick Wm. Hornbeck	8. CONTRACT OR GRANT NUMBER(s): N61339-73-C-0184
9. PERFORMING ORGANIZATION NAME AND ADDRESS: San Diego State University Foundation, 5402 College Avenue, San Diego, CA 92115	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Personnel Research and Development Center, San Diego, CA 92152	12. REPORT DATE: 21 February 1974
	13. NUMBER OF PAGES: ~~25~~ 25
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Naval Training Equipment Center, Orlando, Florida 32813	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer assisted instruction
computer graphics
voice synthesization

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

CAMELOT is a computer-assisted instructional system designed to support computer-generated graphics, computer-generated vocalizations, and computer-selected slide projection as well as full ASCII alphanumeric code. This report consists of a description of the concepts and facilities of CAMELOT, a description of the hardware configuration under development for the Naval Personnel Research and Development Center, San Diego (NPRDC), and the external specifications of the ...

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. author language GRAIL (Graphics Assisted Instructional Language). GRAIL is a rich language in comparison to other CAI languages--allowing a maximum of flexibility in strategies and procedures. The inclusion of a macro definition capability allows for efficiency and ease of coding. The GRAIL compiler is written in IBM 360 Assembly Language. CAMMAC can be readily transported to any IBM 360 or 370 series machine. Complete documentation will be included in a CAMAC Task Report which should be available in the summer of 1974.

Contract N61339-73-C-0184

Quarterly Technical Report

October 1, 1973 - December 31, 1973

Project CAMELOT: Preliminary Report on a CAI System

Project CAMELOT: Preliminary Report on a CAI System

Lynn H. Brock and Frederick Wm. Hornbeck

San Diego State University

While some effort has been expended on the other items on this contract, the most substantial effort and most significant accomplishments have all involved contract item 4. This item entails the design and delivery of hardware and software components of a computer system to support instructional activities incorporating computer-generated graphic displays, computer-generated vocalizations, and program-selected slides from a random access slide projector. The software also provides for the collection of data specified by courseware authors. This report contains a description of the concepts and facilities incorporated in the software package, a description of the hardware configuration, and the external specifications of the author language in modified BNF notation. Primary responsibility for the design and implementation of this powerful but inexpensive computer-assisted instructional system has been delegated to Mr. Lynn Brock.

Concepts and Facilities

This project was undertaken initially to provide a low-cost graphic display system in support of ongoing research activities at the Naval Personnel Research and Development Center, San Diego (NPRDC). Support of a random-access slide projector is also required for this application. In the early stages of the design process, it was determined that the system could--at relatively little additional cost--also incorporate support of a voice synthesizer. This capability is expected to be of considerable value in the planned research activities of this contractor as well as others at NPRDC.

Having established GRAIL as the acronym for the author language (GRaphic Assisted Instructional Language) of this CAI system, the overall project acquired the name CAMELOT. A fully-equipped CAMELOT student station consists of a graphic display terminal, random-access slide projector, and a voice synthesizer. Program authors can select any of these devices for the presentation of information to the student (subject). Students can communicate to the program through the terminal keyboard (full ASCII upper

and lower case character set) and through the positioning of the crosshair cursor on the graphic display screen.

Programs written in GRAIL are compiled and supported during execution by software developed in IBM 360 Assembly language. CAMELOT will be quite portable--subject to easy conversion to any operating system which runs on the IBM 360-370 series of computers. Some of the noteworthy features of these two CAMELOT subsystems are listed below.

GRAIL: Graphic Assisted Instructional Language

GRAIL is the 'author language' in the CAMELOT system and provides the following features:

(1) Display of alphanumeric and/or graphic information. Upper and lower case alphanumerics may be mixed (along with graphic information) in the same display.

(2) Selection and display of any one of eighty slides located in the Random Access Slide Projector.

(3) Vocalization of phonemically encoded messages. Messages may consist of phonemes, words (from the dictionary file), or larger textual passages.

(4) Control of the sequence of presentation and analysis, both on logical condition testing and iteratively. Logical combinations (and, or) of relational (equal to, not equal, less than, etc.) expressions, with grouping (indicated via parenthesis) is allowed.

(5) Input from the terminal includes upper and lower case alphanumeric as well as the current position of the crosshair cursor (upon instruction).

(6) A significant level of 'compile time' checking including
   (a) checks for proper 'nesting' of constructs (i.e., if one construct must be wholly contained within another then an error message results if the author violates the restriction).
   (b) checks for proper data types (e.g., a character string variable used in an arithmetic expression is invalid).

Although the above features are almost always encountered in CAI author languages, the following are relatively unique:

(7) Re-entrant code, which allows several students to be 'taught' using the same machine instructions, thereby saving storage.

(8) Expressions (character and arithmetic) are allowed virtually anywhere a variable can occur.

(9) Extensible language facility. The Macro Definition Language (MDL) is an integral part of GRAIL and allows additional operations and

constructs to be added to the language by the author. The MDL allows the author to have virtually complete freedom in the selection of basic operations or strategies to be used in teaching a course. Facilities in the MDL allow the author to add new commands to the language and examine the operands coded for each use of such operands. As a result of examining those operands, and additionally on the basis of information retained from previous users of the same or other commands, the author has complete control over the resulting operations, and can repetitively generate the same operation (2), or any sequence of operations as desired.

(10) Dynamic storage allocation. Main (core) storage is allocated only when the structure requiring it is entered, thus reducing main storage use.

(11) Portability. All operating system dependent code is isolated into one area, allowing easy conversion to any operating system which runs on the IBM 360-370 series of computers.

(12) Generation of machine code. Actual 'machine code' (computer instructions) is generated by GRAIL. This is typically 5 to 10 times faster in execution than 'interpretive' systems.

EXCALIBUR: Extract and Collect Logically Indicated Basic User Records

EXCALIBUR provides for the collection, maintenance and selection of author selected information with the following features.

(1) Author controlled recording at any time, in addition to recording student logon, logoff times.

(2) FORTRAN (as well as PL/I, COBOL and ASSEMBLER) compatible records, allowing data analysis to be done on any machine supporting magnetic tape and FORTRAN.

(3) Easy selection of records for further analysis through very simple PL/I programs.

Hardware

In our development of CAMELOT facilities, we have used established standards (e.g. ASCII character code) and industry conventions (e.g. RS-232-C hardware interfaces) wherever possible. Consequently, CAMELOT can be implemented in a variety of configurations other than that described here. In particular, GRAIL-programs can be used with terminals other than those we are using.

Student Station

Hardware costs for graphic display capability have been kept at low level through use of a storage-tube device--the Tektronix 4012 Computer Display Terminal. Software is being developed for support of the somewhat more expensive but extremely versatile Tektronix 4014 Computer Display Terminal. With its enhanced graphics option, the 4014

(1) has a display area 11 inches high by 15 inches wide,

(2) has the full ASCII character set (94 printing characters),

(3) has four program-selectable character sizes (from 74 characters per line with 35 lines per display to 133 characters per line with 64 lines per display),

(4) has a vector drawing time of 4000 inches/second,

(5) has--in discrete plot mode--a resolution of 4096 by 4096 addressable points,

(6) has--in interactive mode--1021 points horizontally and 780 points vertically addressable by the thumbwheel controlled crosshair cursor; viewing completely free from parallax,

(7) depending on option selected--either strap selectable or switch selectable input/output data rates of 110, 150, 300, 600, 1200, 2400, 4800 and 9600 bits/second,

(8) has limited vector refresh capability (which allows the temporary display of a limited number of vectors--only one or two at the 1200 band data transmission rate--on the otherwise permanent display), and

(9) has gray scale capability.

Permanent hard copies of terminal displays can be obtained by attaching a Tektronix 4610 hard copy unit to the terminal. Hard copy generation may be initiated by a switch on the terminal keyboard, a switch on the hard copy unit, or by computer command. An optional multiplexor allows a single hard copy unit to service up to four terminals.

Richer visual displays are permitted through the inclusion of a Kodak RA-960 random access slide projector in the student station configuration. Any of the 80 slides in the carousel can be selected and displayed under program control. The projector can also be turned on or off by the computer.

The presentation of computer-generated vocalizations is accomplished through a Federal Screw Works Votrax (Model 5.1) voice synthesizer. This inexpensive device allows the programmer to select the auditory mode of communication for such purposes as addressing preliterate or blind students, conveying additional information without disturbing the visual display, providing instruction in phonics, or simply to provide emphasis through use of an alternate medium. The

VOTRAX provides great flexibility in the presentation of vocal information, since it has an unlimited vocabulary of words, and has none of the mechanical timing peculiarities of cassette tapes or disks.

A purchase order has been placed with Sensors, Data, Decisions, Inc. of San Diego for the design, fabrication, and installation of the Student Station Interface (SSI). This device will pass all eight-bit codes except those specified for device selection. Device selection codes will switch the computer output data stream to the appropriate student station device. (Even the device selection codes can be passed through the SSI using special notation.)

Computer

Our present system configuration is designed to support a maximum of four simultaneous CAMELOT users. We expect to be able to accomplish this with a dedicated area of 64K bytes on our IBM System 360/40 computer. Support of a larger number of simultaneous users would require a larger dedicated area and/or the implementation of a timesharing operating system. (We now run under IBM's DOS)

Disk storage utilization will depend on the amount of courseware to be supported and the number of students to use the system. Student records will be dumped to magnetic tape on a regularly scheduled basis--probably daily.

Communications

Because of the small number of terminals in the present configuration, it has not been necessary to resort to multiplexing to obtain cost effectiveness. All communications will be by voice-grade lines on the direct dial network at 1200 baud. Interfacing at the terminal end is via a Bell System 1000A Data Access Arrangement and a General DataComm 202-9A asynchronous modem. At the computer end, a Data General NOVA 1220 mini computer (on the main computer's multiplexor channel) handles intra-line editing of terminal-generated input to the computer. The phone line--NOVA interface is accomplished via a Bell System 1001A Data Access Arrangement and a General DataComm 202-9A a-synchronous modem.

Hardware Configuration Restraints

CAMELOT is suitable for installation on a wide variety of hardware configurations but there are two constraints aside from obvious size considerations (i.e. more terminals require more core in both the NOVA and the 360). First, the system is easily transportable only to configurations in which the main computer is of the IBM 360-370 series. Second, communications rates of at least 1200 baud are desirable for any significant use of graphics. Rates up to 9600 baud can be maintained by all components of the configuration described here except the voice synthesizers, modems, and telephone lines. Higher rates could be supported by locating the terminals in the immediate proximity of the computer to allow hard wiring, by

utilization of conditioned lines in the direct dial system, or by use of microwave or other high speed communications equipment.

GRAIL Language Specifications

GRAIL has been designed to provide maximum flexibility to the courseware author-programmer. This is an essential characteristic for a language developed specifically for research and development applications. At the same time, however, one can achieve the simplicity of other CAI author languages (e.g. PLANIT or PLATO's TUTOR) through use of the Macro Definition Language (MDL) which has been incorporated in GRAIL. The presence of MDL should facilitate the simulation or emulation of other CAI systems (e.g. TICCIT) as well as the development of innovative or experimental procedures or strategies. The external specifications for GRAIL instructions follow. (In the interests of economy, we have simply copied a printout of our online version of the specifications rather than have them retyped.)

****

EXTERNAL DOCUMENTATION INDEX

****

****

NOTATION

****

NOTATION

SYNTACTIC VARIABLES:
ANYTHING BEGINNING WITH THE CHARACTER '#' IS REPLACED BY THE FORM INDICATED FOR THE VARIABLE UNDER 'DEFINITIONS'.

METASYMBOLS:
THE CHARACTERS ' (APOSTROPHE), '<', '>', AND '/' ARE TO BE TREATED AS METASYMBOLS UNLESS THEY ARE ENCLOSED IN APOSTROPHES. THEY HAVE THE FOLLOWING MEANINGS:

<,> ARE USED TO INDICATE THAT THE ENCLOSED FORM IS OPTIONAL AND MAY BE OMITTED AT THE USER'S DISCRETION.

/ INDICATES THAT A CHOICE MUST BE MADE AMONG THE ALTERNATIVE FORMS WHICH IT SEPARATES; IF A DEAULT IS ALLOWED, IT PRECEEDS THE FIRST /, UNLESS OTHERWISE INDICATED BY THE TEXT.

THE FORM ',...' INDICATES THAT THE PRECEEDING FORM MAY BE REPEATED A NUMBER OF TIMES (AS SPECIFIED IN THE TEXT) WITH EACH FORM SEPARATED BY COMMAS.

THE FORM '...' INDICATES THAT THE PRECEEDING SYNTACTIC VARIABLE MAY BE REPEATED A NUMBER OF TIMES.

PARENTHESIS '(' AND ')' ALWAYS INDICATE THAT ACTUAL PARENTHESES OCCUR, AND IN THE FORM <(> FORM <)>, EITHER BOTH PARENTHESES MUST BE PRESENT OR BOTH OMITTED.

APOSTROPHES INDICATE THAT THE ENCLOSED CHARACTER OR CHARACTERS ARE TO BE TREATED AS THEMSELVES AND NOT AS METASYMBOLS; THUS, THE FORM '' INDICATES AN ACTUAL 'SINGLE' APOSTROPHE - ' AND '/' IS

****

LANGUAGE SPECIFICATIONS

****

@LABEL FRAME

STORAGE IS ALLOCATED FOR VARIABLES DEFINED IN THE FRAME AND $FRAME IS SET TO @LABEL.

FEND @MLABEL<,ERASE=YES / NO>

ENDF DEFINES THE END OF A FRAME , CAUSING STORAGE DEFINED FOR THE FRAME TO BE RELEASED. IF THE ERASE=NO OPTION IS SPECFIED THEN THE SCREEN IS NOT ERASED , OTHERWISE IT IS ERASED.

@OLABEL TEXT (<'@TEXT-STRING' / @CEXP >,...),<POSIT=(<@LINENO><,@POS>)>

<,CRLF=YES / NO>

CAUSES THE INDICATED STRINGS OF CHARACTERS TO BE DISPLAYED. IF EITHER @LINENO OR @POS IS SPECIFIED THE STRING WILL BE DISPLAYED AT THE CURRENT ALPHA-CURSOR POSITION.

IF THE POSIT PARAMETER IS SPECIFIED , @LINENO INDICATES THE LINE WHERE THE DISPLAY IS TO BEGIN AND @POS THE POSITION WITHIN THE LINE WHERE THE FIRST CHARACTER IS TO BE DISPLAYED. IF @LINENO IS OMITTED , THE CURRENT LINE IS ASSUMED AND IF @POS IS OMITTED IT IS ASSUMED TO BE 1.

REGARDLESS OF THE LINE PARAMETER (OR IN IT'S ABSENCE) THE ALPHA-CURSOR IS MOVED TO THE FIRST CHARACTER OF THE NEXT LINE AFTER THE CHARACTERS ARE DISPLAYED , UNLESS THE CRLF=NO PARAMETER IS USED , IN WHICH CASE THE ALPHA-CURSOR WILL BE LEFT AT THE POSITION IMMEDIATELY FOLLOWING THE LAST CHARACTER DISPLAYED.

@OLABEL SHOW (@FEXP)

THE SLIDE SPECIFIED BY @FEXP IS SELECTED AND DISPLAYED ON THE RANDOM ACCESS SLIDE PROJECTOR. IF @FEXP IS GREATER THAN THE MAXIMUM VALID SLIDE NUMBER FOR THE TERMINAL THEN THE COMMAND IS IGNORED. IF THE TERMINAL DOES NOT HAVE A SLIDE PROJECTOR (AS INDICATED BY THE EXECUTION VARIABLE $FRASP) THEN THE COMMAND IS IGNORED.

@OLABEL FCALL @FORNAME,(@EXP-1,EXP-2,EXP-3,...)

THE FORTRAN SUBROUTINE IDENTIFED BY @FORNAME IS CALLED ,

WITH THE ARGUMENTS SPECIFIED BY @EXP-1,@EXP-2...
THE SUBROUTINE CALLED MUST BE KNOWN TO THE SYSTEM OR
OR AN ERROR MESSAGE WILL RESULT DURING COMPILATION
IT IS THE USER'S RESPONSIBILITY
TO ENSURE THAT THE TYPES OF THE ARGUMENTS PASSED
AGREE WITH THE TYPES EXPECTED BY THE SUBROUTINE.

@OLABEL EXEC (@CEXP)

THE SECTION IDENTIFIED BY THE FIRST 6 CHARACTERS
OF THE RESULT OF EVALUATING @CEXP IS EXECUTED ,
AFTER WHICH CONTROL RETURNS TO THE NEXT STATEMENT

IF THE RESULT IS LESS THAN SIX CHARACTERS , THEN IT IS PADDED WITH TRAILING BLANKS. THIS STATEMENT MAY OCCUR ONLY IN A COURSE COMPILE UNIT.

@LABEL COURSE

DEFINES A COURSE WITH THE INDICATED NAME , CAUSES STORAGE TO BE ALLOCATED FOR VARIABLES DEFINED IN THE COURSE AND THE SCREEN TO BE ERASED. $COURSE IS SET TO @LABEL AND $FRAME AND $SECTION ARE SET TO NULL .

*

CRSEND @MLABEL

DEFINES THE END OF A COURSE AND CAUSES ALL STORAGE ALLOCATED FOR THE COURSE TO BE RELEASED.

```
@LABEL   IF     (@CON-CLS)
         THEN   <NULL>
         ELSE   <NULL>
         IEND   @MLABEL
```

THESE FOUR OPERATIONS ALLOW FOR THE CONDITIONAL EXECUTION OF SUBSEQUENT STATEMENTS AS FOLLOWS -

1. IF @CON-CLS IS TRUE , THE STATEMENTS BETWEEN THE THEN AND ELSE OPERATIONS ARE EXECUTED , AFTER WHICH CONTROL PASSES TO THE STATEMENT FOLLOWING THE IEND

2. IF @CON-CLS IS FALSE , THE STATEMENTS BETWEEN THE ELSE AND IEND ARE EXECUTED , AFTER WHICH CONTROL PASSES TO THE STATEMENT FOLLOWING THE IEND

3. NO STATEMENTS MAY OCCUR BETWEEN THE IF AND THEN OPERATIONS.

4. IF THE 'NULL' OPTION IS SPECIFIED FOR A THEN OR ELSE OPERATION , AND IF CONTROL IS PASSED TO IT BY THE IF OPERATION , THE CONTROL IMMEDIATELY PASSES TO THE STATEMENT FOLLOWING THE IEND.

5. IF CONSTRUCTS MAY BE 'NESTED' , THAT IS - AN IF CONSTRUCT MAY OCCUR BETWEEN THE THEN AND ELSE , OR ELSE AND IEND , OPERATIONS OF ANOTHER IF CONSTRUCT.

6. ALL FOUR OPERATIONS COMPRISING THE IF CONSTRUCT ARE REQUIRED WHEN THE CONSTRUCT IS USED.

```
@LABEL   DO     WHILE / UNTIL (@CON-CLS)
         < STATEMENTS >
         DEND   @MLABEL
```

THESE TWO OPERATIONS ALLOW FOR THE REPETITIVE EXECUTION OF A GROUP OF STATEMENTS AS FOLLOWS -

```
                WHEN CONTROL ENTERS THE DO STATEMENT , THE VALUE OF
              @CON-CLS IS TESTED. IF IT IS TRUE (FALSE) AND THE
              WHILE (UNTIL) FORM WAS USED , THEN THE STATEMENTS
              BETWEEN THE DO AND DEND ARE EXECUTED , AFTER WHICH
              CONTROL AGAIN ENTERS THE DO STATEMENT.
              WHENEVER CONTROL ENTERS THE DO STATEMENT AND
              @CON-CLS IS FALSE (TRUE) AND THE WHILE (UNTIL) FORM WAS
              USED , THEN CONTROL PASSES TO THE STATEMENT
              FOLLOWING THE DEND.
*
@LABEL    DOINC @FVAR,@FEXP-1,TO,@FEXP-2<,BY,@FEXP-3>
          DODEC
          < STATEMENTS >
          DEND  @MLABEL
                WHEN CONTROL ENTERS THE DOINC (DODEC) STATEMENT
                @FEXP-1 IS EVALUATED AND ASSIGNED TO @FVAR;
                @FEXP-2 IS EVALUATED AND SAVED;
                @FEXP-3 IS EVALUATED AND SAVED (IF @FEXP-3
                IS OMITTED IT IS ASSUMED EQUAL TO ONE).
                @FVAR IS NOW COMPARED TO @FEXP-2 , AND IF @FVAR IS
                GREATER THAN (LESS THAN) @FEXP-2 THEN CONTROL PASSES TO
                THE STATEMENT FOLLOWING THE DEND. IF @FVAR IS LESS
                THAN OR EQUAL (GREATER THAN OR EQUAL) TO
                @FEXP-2 THEN THE STATEMENTS
                BETWEEN THE DOINC (DODEC) AND THE DEND STATEMENT ARE
                EXECUTED. WHEN CONTROL ENTERS THE DEND STATEMENT
                THE SAVED EVALUATION OF @FEXP-3 IS ADDED TO
                (SUBTRACTED FROM) @FVAR. CONTROL IS THEN TRANSFERED
                TO THE POINT IN THE DOINC (DODEC) WHERE @FVAR IS
                COMPARED TO @FEXP-2 AND THE ABOVE PROCESS IS
                REPEATED.


@LABEL    CASES @FLIT<,EXEC=FIRST / ALL>
          CASE  (@CON-CLS-1)
          < STATEMENTS >
          CASE  (@CON-CLS-2)
          < STATEMENTS >
          CASE
          .
          .
          .
          CASE  (@CON-CLS-@FLIT)
          < STATEMENTS >
          CEND  @MLABEL
                THE CASES CONSTRUCT ALLOWS SELECTION OF GROUPS
                OF STATEMENTS TO BE EXECUTED BASED UPON THE THE
                LOGICAL VALUES OF CONDITIONAL CLAUSES. WHEN THE
                CASES STATEMENT IS ENTERED THE @CON-CLS OF EACH
                CASE STATEMENT IN THE CONSTRUCT IS EVALUTED ,
                BEGINNING WITH THE FIRST. IF THE @CON-CLS OF
                ANY PARTICULAR CASE STATEMENT IS TRUE THEN THE
                STATEMENTS BETWEEN THAT CASE STATEMENT AND THE
                NEXT CASE [illegible] STATEMENT OF THE CONSTRUCT
```

ARE EXECUTED. IF THE EXEC=ALL OPTION IS SPECIFIED
THEN THE EVALUATION OF EACH @CON-CLS CONTINUES;
IF THE EXEC EXEC=FIRST OPTION IS SPECFIED THEN
ONLY THE STATEMENTS BETWEEN THE FIRST CASE
STATEMENT WHICH EVALUATED TO TRUE AND THE NEXT
CASE OR CEND STATEMENT WILL BE EXECUTED.
WITH EITHER EXEC OPTION AT LEAST ONE @CON-CLS
MUST BE TRUE OR EXECUTION OF THE COURSE WILL
BE TERMINATED WITH AN ERROR.
@FLIT IS USED TO SPECIFY THE NUMBER OF CASE
STATEMENTS WITHIN THE CURRENT CASES CONSTRUCT AND
MUST BE EXACTLY EQUAL TO THE NUMBER OF CASE
STATEMENTS.

@LABEL READ <@CVAR><,POSIT=(<@LINENO><,@POS>)>
CAUSES CHARACTERS TO BE READ FROM THE TERMINAL INTO
@CVAR. IF MORE CHARACTERS ARE ENTERED THAN @CVAR CAN
HOLD , THEY ARE LOST. IF @CVAR IS OMITTED THEN THE
EXECUTION VARIABLE $ANSWER IS USED AND HAS A MAXIMUM
LENGTH OF 72. IF THE LINE OPTION IS SPECIFIED THEN THE
ALPHA-CURSOR IS MOVED TO THE INDICATED POSITION ON THE
SCREEN BEFORE INPUT IS ACCEPTED (SEE 'TEXT').

@LABEL CALC @VAR,(@EXP)
@EXP IS EVALUATED AND ASSIGNED TO @VAR. @EXP AND @VAR
MUST BE OF THE SAME TYPE. IF A CHARACTER STRING
ASSIGNMENT IS PERFORMED AND TRUNCATION OCCURS THEN
THE EXECUTION VARIABLE $FTRUNC WILL BE SET
TO THE NUMBER OF CHARACTERS TRUNCATED , OTHERWISE
IT WILL BE SET TO ZERO.

@LABEL READCC <@FVAR-1><,@FVAR-2><,@CVAR>
THE CROSSHAIR CURSOR IS ILLUMINATED AND IT'S X-Y
COORDINATES ARE READ WHEN THE NEXT CHARACTER IS TYPED
BY THE USER. THE X VALUE , THE Y VALUE AND THE
CHARACTER TYPED ARE READ INTO @FVAR-1,@FVAR-2 AND
@CVAR RESPECTIVELY. IF ANY (OR ALL) OF THE OPERANDS
ARE OMITTED , THE VALUES ARE READ INTO THE EXECUTION
VARIABLES $FCCX , $FCCY AND $CCHR RESPECTIVELY.
NOTE THAT THE X AND Y VALUES READ ARE IN SCREEN
COORDINATES AND THAT THE RESULTING LENGTH OF @CVAR
(OR $CCHR) WILL ALWAYYS BE ONE.

@LABEL RECORD <@CEXP-1,@CEXP-2,...,>TYPE=(@CEXP)
@CEXP-1,@CEXP-2,... ARE CONCATENATED AND WRITTEN
ON THE RECORDER FILE. THE RECORD IS IDENTIFIED BY THE
FIRST TWO CHARACTERS OF THE CHARACTER STRING RESULTING
RESULTING FROM EVALUATION OF THE @CEXP SPECIFIED
FOR TYPE. IT IS THE USER'S RESPONSIBILITY TO ENSURE
THAT THE RESULTING RECORD HAS THE CORRECT FORMAT FOR
IT'S TYPE , AND THAT THE TYPE ITSELF IS VALID.

@LABEL DELAY *DELAY

EXECUTION OF SUBSEQUENT OPERATIONS IS DELAYED BY THE SPECIFIED NUMBER OF SECONDS.

@LABEL ESCAPE @MLABEL
THIS COMMAND ALLOWS EXIT FROM WITHIN A NEST OF 'IF' , 'DO' , 'DODEC' , DOINC' AND/OR 'CASES' CONSTRUCTS. WHEN CONTROL ENTERS THE ESCAPE STATEMENT THE DEND,IEND OR CEND HAVING THE MATCHING @MLABEL WILL BE THE NEXT STATEMENT EXECUTED.
NOTE THAT THE STATEMENT HAVING THE MATCHING @MLABEL *MUST* BE WITHIN THE ACTIVE NEST AT THE TIME THE ESCAPE STATEMENT IS EXECUTED.

LIMIT @FLIT
THE LIMIT STATEMENT IS USED TO SPECIFY AN UPPER BOUND ON HOW MANY TIMES THE STATEMENTS WITHIN A 'DO' , 'DODEC' OR 'DOINC' MAY BE EXECUTED.

IF NO LIMIT STATEMENT IS PRESENT THEN EACH SUCH CONSTRUCT IS LIMITED TO 100 REPETITIONS. IF A LIMIT STATEMENT IS PRESENT , THEN THE NEXT (AND *ONLY* THE NEXT) 'DO' , 'DOINC' OR 'DODEC' CONSTRUCT IN THE SOURCE PROGRAM WILL BE LIMITED TO @FLIT REPETITIONS. @FLIT MAY BE LESS THAN 100 BUT MUST BE GREATER THAN ZERO.

@OLABEL ERASE
THE SCREEN IS ERASED. IF THE TERMINAL IS NOT CAPABLE OF BEING ERASED THEN THE STATEMENT HAS NO EFFECT.

VTYPE C(@L-1,@U-1),F(@L-2,@U-2),E(@L-3,@U-3)
THIS OPERATION IS USED TO INDICATE THAT ANY VARIABLE OR FUNCTION REFERENCE IN SUBSEQUENT STATEMENTS WILL HAVE IT'S TYPE DETERMINED AS FOLLOWS-
IF THE FIRST CHARACTER OF THE VARIABLE OR FUNCTION IS IN THE RANGE @L-N,@U-N THEN THE TYPE IS ASSUMED TO BE INDICATED BY THE CHARACTER PRECEEDING THE PARENTHESIS IN WHICH THE RANGE WAS ENCLOSED.
IF THE FIRST CHARACTER OF THE VARIABLE OR FUNCTION DOES NOT FALL IN ANY OF THE RANGES , THEN THE TYPE IS ASSUMED TO BE CHARACTER.
AT THE TIME THE VTYPE STATEMENT IS ENCOUNTERED THE FOLLOWING ITEMS ARE CHECKED-
1. THE RANGES MUST BE DISJOINT IE. THE RANGES MUST NOT OVERLAP.
2. @L-N MUST BE LESS THAN OR EQUAL TO @U-N FOR EACH RANGE
3. ONLY ONE VTYPE STATEMENT IS ALLOWED PER COMPILE UNIT , AND IT MUST IMMEDIATELY FOLLOW THE COURSE OR SECTION STATEMENT FOR THE COMPILE UNIT.

VDEF @VAR-1<#@MAXLEN>,@VAR-2<#@MAXLEN>...
@VAR-1,@VAR-2... ARE DEFINED AND RESERVED STORAGE BY THEIR APPEARANCE IN THE VDEF STATEMENT. THE TYPE (C,F OR E) OF EACH VARIABLE IS DETERMINED

BY THE FIRST CHARACTER OF IT'S NAME IN CONJUNCTION WITH THE VTYPE STATEMENT CURRENTLY IN FORCE. IF THE LEVEL=COURSE OPTION IS USED THEN THE STORAGE IS MERELY DESCRIBED AND NO STORAGE IS RESERVED , SINCE THE STORAGE WILL BE ALLOCATED BY THE COURSE WHICH EXECUTES THE SECTION.
IF THE LEVEL=SECTION OPTION IS USED , THEN THE STORAGE IS DEFINED FOR ALL FRAMES IN THE CURRENT SECTION.
IF THE LEVEL OPTION IS NOT USED THEN THE VARIABLES ARE DEFINED (AND MAY BE USED) ONLY AT THE CURRENT LEVEL.
A NAME MAY APPEAR IN ONLY ONE VDEF OR CDEF STATEMENT WITHIN THE COMPILE UNIT.

CDEF @VAR-1(@LIT),@VAR-2(@LIT)...
@VAR-1,@VAR-2,...ARE DEFINED , RESERVED STORAGE AND SET TO THE VALUE INDICATED BY THE ASSOCIATED @LIT. QUANTITIES DEFINED IN THIS WAY MUST NOT HAVE THEIR VALUE CHANGED DURING EXECUTION. MORE EFFICIENT EXECUTION WILL RESULT IF THE NUMBER OF CDEF STATEMENTS IS KEPT TO A MINIMUM.
ALL OTHER RULES WHICH APPLY TO VARIABLES DEFINED BY THE VDEF STATEMENT APPLY TO THE CDEF STATEMENT , WITH THE EXCEPTION THAT THE LENGTH OF CHARACTER VARIABLES IS DETERMINED BY THE LENGTH OF THE ASSOCIATED @LIT.

$CFRAM- 1 TO 6 CHARACTER VARIABLE
CONTAINS THE NAME OF THE CURRENT FRAME DURING EXECUTION.

AT THE COURSE LEVEL , IT CONTAINS THE NAME OF THE MOST RECENTLY EXECUTED FRAME.

$FLINE- INTEGER
GIVES THE CURRENT LINE NUMBER OF THE ALPHA-CURSOR

$FPOS- INTEGER GE
GIVES THE CURRENT POSITION , WITHIN THE LINE , OF THE ALPHA-CURSOR.

$CDATE- 8 CHARACTER VARIABLE
CONTAINS THE CURRENT DATE IN THE FORM MM/DD/YY.

$CDOW- 6 TO 9 CHARACTER VARIABLE CONTAINING THE CURRENT DAY OF THE WEEK IE.-MONDAY,TUESDAY,...

$CMNTH- 3 TO 9 CHARACTER VARIABLE CONTAINING THE CURRENT MONTH IE. JANUARY,MAY,...

$CFNAM- 0 TO 40 CHARACTER VARIABLE
CONTAINS THE FIRST NAME OF THE STUDENT CURRENTLY RUNNING.

$CLNAM- TO 40 CHARACTER VARIABLE
CONTAINS THE LAST NAME OF THE STUDENT CURRENTLY RUNNING.

$FIDNUM - 9 CHARACTER VARIABLE
CONTAINS THE ID NUMBER OF THE STUDENT CURRENTLY RUNNING.

$COURSE- 1 TO 6 CHARACTER VARIABLE
CONTAINS THE NAME OF THE COURSE CURRENTLY RUNNING.

```
**********************************************************************
                SYNTACTIC VARIABLES
**********************************************************************

      @CON-CLS := @REXP / <(> @REXP:@LOP:@REXP <)>
      @ROP  :=   EQ / NE / GT / LT / GE / LE / NL / NG
      @LOP  :=   OR / AND
      @AEXP :=   @FEXP / @EEXP
      @FEXP := ( @FVAR / @FREF / @FLIT / <(> @FEXP@AOP@FEXP <)> )
      @EEXP := ( @EVAR / @FREF / @ELIT / <(> @EEXP@AOP@EEXP <)> )
      @AOP  :=   + / - / * / '//'  (// IS THE REMAINDER OPERATOR)
      @CEXP := ( @CVAR / @CLIT / @FREF / @CEXP@COP@CEXP )
      @COP  :=   ;
      @FREF := @FNAME ( @EXP,@EXP,... )
      @VAR  := @CVAR / @FVAR / @EVAR
      @CVAR , @FVAR , @EVAR := 1 TO 6 ALPHANUMERIC CHARACTERS ,
            THE FIRST OF WHICH MUST BE ALPHABETIC.
      @LIT := @FLIT / @ELIT / @CLIT
      @CLIT := '''@STRING'''
      @FLIT := @S@UFLIT
      @UFLIT := @D@UFLIT
      @ELIT := @FLIT@P<@UFLIT><E@FLIT>
      @S := '' / + / -
      @D := 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9
      @P := .
      @STRING := @C / @STRING@C
      @C := A / B / C /.../ Z / @D /
            @AOP / @COP / '.' / ''' / & / ? / $ /
            # / ( / ) / < / > / @P / ! / =

      @LABEL := @A / @A@LABEL / @A@D
            REQUIRED LABEL , WHICH MUST BE 6 OR FEWER
            CHARACTERS LONG.

      @TEXT-STRING := @STRING
            CERTAIN CHARACTERS WILL CAUSE THE EFFECTS NOTED
            NOTED BELOW IF PRESENT IN @STRING -
            @ - AT SIGN
                  INDICATES THAT THE SINGLE LETTER IMMEDIATELY
                  FOLLOWING IT IS TO BE DISPLAYED AS UPPER CASE.
            @@- TWO AT SIGNS
                  INDICATES THAT ALL LETTERS BETWEEN THE @'S
                  AND THE NEXT @ ARE TO BE DISPLAYED AS UPPER CASE.
            _ - UNDERSCORE
                  INDICATES THAT THE TEXT BETWEEN IT AND THE NEXT
                  UNDERSCORE IS TO BE UNDERLINED.

            ANY APOSTROPHES (') OR AMPERSANDS (&) WHICH ARE
            TO BE DISPLAYED MUST BE PRESENT TWICE.
```

PHONETIC INPUT DATA FORMAT

PHONEMES ARE IDENTIFIED BY THE ONE TO THREE CHARACTER CODE USED IN THE VOTRAX LITERATURE, AND ARE SEPARATED FROM THE INFLECTION BY ONE OR MORE COMMA'S OR BLANKS. THE INFLECTION IS INDICATED BY A SINGLE DIGIT - 1 FOR IN1, 2 FOR IN2, 3 FOR IN3, AND 4 FOR IN4. AN OMITTED INFLECTION IS ASSUMED TO BE 2.

EXAMPLE:

S, 1, AH1, IY, T, 3, R, UH3, 4, AH1, N, 3, IH 1, K, 1, S, 1

IS ENCODED AS HEX

X'9F.DE.C9.2A.CB.63.D5.CD.85.99.9F'

****

ERROR MESSAGES

****

STYP0001 - LITERAL OR VARIABLE NOT KNOWN
STYP0002 - INVALID FIXED POINT LITERAL
STYP0003 - FIXED POINT LITERAL TO LARGE
STYP0004 - INVALID VARIABLE
STYP0005 - INVALID FLOATING POINT LITERAL
STYP0006 - FLOATING POINT LITERAL TO LARGE
STYP0007 - EXPONENT LITERAL ERROR
SNTP0001 - NAME NOT TYPED PER VTYPE STATEMENT
SQOT0001 - IMPROPER USE OF QUOTATION MARKS
SPRN0001 - IMPROPER USE OF PARENTHESES
CDEF0001 - NULL SUBSTRING
CDEF0002 - INVALID QUOTMARK IN NAME FIELD
CDEF0003 - NO VARIABLE OR IMPROPER VARIABLE FIELD
CDEF0004 - LABEL VS. VARIABLE TYPE ERROR
CDEF0005 - NAME TYPE UNKNOWN MADE EQUAL TO LITERAL TYPE
CDEF0006 - VARIABLE LENGTH NOT GREATER THAN 0 CHARACTERS
CDEF0007 - NO LEVEL PARAMETER OR ERROR IN LEVEL PARAMETER
CDEF0008 - CDEF GENERATED AT FRAME DEFAULT LEVEL
CDEF0009 - NAME GREATER THAN 6 CHARACTERS LONG
CDEF0010 - CDEF CODE NOT GENERATED
VDEF0001 - NO LEVEL PARAMETER OR MISSING LEVEL PARAMETER
VDEF0002 - VDEF GENERATED AT FRAME DEFAULT LEVEL
VDEF0003 - NULL SUBSTRING
VDEF0004 - NAME GREATER THAN 6 CHARACTERS LONG
VDEF0005 - NAME TYPE UNKNOWN
VDEF0006 - MAXLINE NOT NUMERIC
VDEF0007 - INVALID MAXLINE NUMBER
VDEF0008 - NAME TYPE VS. MAXLINE LENGTH ERROR
VDEF0009 - CODE NOT GENERATED
VTYP0001 - INVALID PARAMETER LENGTH
VTYP0002 - INVALID PARAMETER FORMAT
VTYP0003 - DUPLICATE PARAMETER
VTYP0004 - INVALID C PARAMETER
VTYP0005 - INVALID F PARAMETER
VTYP0006 - INVALID E PARAMETER
VTYP0007 - PARAMETERS OVERLAP
VTYP0008 - VTYPE NEEDED ALLOWED THIS ONE
VTYP0009 - ONLY ONE CORRECT VTYPE STATEMENT PER RUN

GRAIL FUNCTIONS

$EFRMF(@FARG)
THE INTEGER CONTAINED IN @FARG IS CONVERTED TO AN E-TYPE NUMBER. NOTE THAT THE CONVERSION IS EXACT IF AND ONLY IF @FARG IS LESS THAN (APPROXIMATELY) 8 DECIMAL DIGITS.

$FFRME(@EARG)
THE FLOATING POINT NUMBER CONTAINED IN @EARG IS CONVERTED TO AN INTEGER. THIS CONVERSION IS ACCOMPLISHED BY IGNORING ANY FRACTIONAL PART. IF THE RESULTING INTEGER IS TO LARGE IN VALUE (> 2**31-1 OR < -2**31) THEN THE RESULT IS THE LARGEST INTEGER OF APPROPRIATE SIGN, AND THE EXECUTION VARIABLE $FFRMEI IS SET TO 1. IF THE CONVERSION IS ACCOMPLISHED WITHOUT ERROR $FFRMEI IS SET TO ZERO.

$CFRMF(@FARG)
THE F-TYPE NUMBER IDENTIFIED BY @FARG IS CONVERTED TO A CHARACTER STRING CONTAINING A DECIMAL REPRESENTATION OF THE NUMBER. THE FORM OF THE RESULT IS THE NUMBER, PRECEDED BY A '-' IF NEGATIVE, WITH LEADING ZEROES SUPPRESSED. A ZERO VALUE IS REPRESENTED BY A SINGLE ZERO WITH NO SIGN. A 'NEGATIVE ZERO' IS NOT POSSIBLE.

$FFRMC(@CARG)
THE STRING CONTAINED IN @CARG IS CONVERTED TO AN F-TYPE NUMBER. @CARG MAY CONTAIN A STRING OF THE FORMAT 'B...SD...B...' WHERE THE B'S INDICATE OPTIONAL SPACES, THE S INDICATES AN OPTIONAL SIGN (+ OR -) AND THE D INDICATES DECIMAL DIGITS. IF THE VALUE REPRESENTED BY THE STRING IS OF TOO LARGE A MAGNITUDE TO BE REPRESENTED BY AN F-TYPE NUMBER, THEN THE RESULT IS THE LARGEST NUMBER OF APPROPRIATE SIGN AND $FFRMCI IS SET TO -1 OR -2 DEPENDING ON WHETHER THE SIGN OF THE RESULT IS POSITIVE OR NEGATIVE, RESPECTIVELY. IF THE STRING IS NOT OF THE CORRECT FORM, THEN A RESULT OF ZERO IS RETURNED AND THE EXECUTION VARIABLE $FFRMCI IS SET TO THE POSITION OF THE CHARACTER IN @CARG WHERE THE ERROR WAS DETECTED.
IF NO ERRORS OCCUR, $FFRMCI IS SET TO ZERO.

$CFRME(@EARG<,@FARG>)
@EARG IS CONVERTED TO A CHARACTER STRING ACCORDING TO THE FOLLOWING RULES-
IF @FARG IS NOT SPECIFIED THEN-
IF THE ABSOLUTE VALUE OF @EARG IS LESS THAN OR EQUAL TO 9999.9999 AND GREATER THAN OR EQUAL TO .0001 THEN THE RESULTING CHARACTER STRING HAS THE FORM 'ZZZZ.9999' WHERE THE Z'S INDICATE ZERO SUPPRESSED POSITIONS (WHICH ARE NOT RETURNED IF THEY CONTAIN SPACES),

AND THE 9'S INDICATE POSITIONS WHICH WILL BE ZERO FILLED UNTIL THE LAST NON-ZERO DIGIT IS ENCOUNTERED. IF THE NUMBER IS NEGATIVE, THEN A '-' IS INSERTED PRIOR TO THE FIRST CHARACTER OF THE RESULTING STRING.

IF THE ABOVE CONDITION IS NOT MET THEN THE RESULTING STRING WILL HAVE THE FORM 'ZZZZZZZ.9999999ESXX' WHERE THE Z'S AND 9'S ARE HANDLED AS ABOVE, AS IS THE SIGN, THE 'E' IS AN ACTUAL CHARACTER IN THE RESULTING STRING AND THE 'S' IS THE SIGN (+ OR -) OF THE EXPONENT (WHICH IS REPRESENTED BY THE 'XX'). THE MEANING OF THIS FORM IS THAT THE NUMBER PRECEEDING THE 'E'

SHOULD BE MULTIPLIED BY TEN RAISED TO THE XX'TH POWER TO PROVIDE THE CORRECT VALUE. THUS THE FORM IS VERY SIMILAR TO STANDARD SCIENTIFIC NOTATION.

IF @FARG IS SPECIFIED THEN CONVERSION IS AS ABOVE EXCEPT THAT THE NUMBER OF DIGITS PRINTED AFTER THE DECIMAL POINT IS EQUAL TO @FARG. HOWEVER, IN NO CASE WILL MORE THAN 7 SIGNIFICANT DIGITS BE GENERATED, SINCE THIS IS THE PRECISION LIMIT OF THE COMPUTER.

IF THE RESULTING STRING IS LONGER THAN THE OUTPUT CHARACTER STRING INTO WHICH IT IS BEING STORED, THEN $EFRMEI (A GLOBAL FULL WORD VARIABLE) IS SET TO -1.

$EFRMC(@CARG)

THE CHARACTER STRING IN @CARG IS CONVERTED TO AN E-TYPE NUMBER ACCORDING TO THE FOLLOWING RULES -
THE STRING MUST BE OF THE FORM 'B...SD...PD...B...ESX...B...'
WHERE THE B'S INDICATE OPTIONAL BLANKS,
THE FIRST 'S' INDICATES AN OPTIONAL SIGN (+ OR -),
THE D'S INDICATE DECIMAL DIGITS TO BE INTERPRETED AS THE SIGNIFICANT DIGITS OF THE NUMBER,
THE 'P' INDICATES AN OPTIONAL DECIMAL POINT,
THE SECOND 'S' INDICATES THE SIGN OF THE EXPONENT AND IS OPTIONAL,
THE 'E' INDICATES THE LETTER E, WHICH IS REQUIRED IF AN EXPONENT IS SPECIFIED,
AND THE X'S INDICATE THE EXPONENT WHICH IS ALSO OPTIONAL.

THE FOLLOWING ADDITIONAL RULES APPLY -

1. IF @CARG IS A NULL STRING, THEN $EFRMCI IS SET TO -1.
2. IF THE STRING REPRESENTS A NUMBER TOO LARGE IN MAGNITUDE TO BE STORED IN AN E-TYPE NUMBER, THEN $EFRMCI IS SET TO -2 AND THE RESULT IS SET EQUAL TO THE LARGEST NUMBER OF APPROPRIATE SIGN.
3. IF THE STRING REPRESENTS A NUMBER TOO SMALL TO BE STORED IN AN E-TYPE NUMBER, THEN $EFRMCI IS SET TO -3 AND THE RESULT IS THE SMALLEST NON-ZERO NUMBER OF APPROPRIATE SIGN.
4. IF ANY OTHER FORMAT ERROR IS ENCOUNTERED, THEN $EFRMCI IS SET TO THE POSITION IN @CARG WHERE THE ERROR WAS DISCOVERED AND THE RESULT IS ZERO.
5. IF NO ERRORS OCCUR, THEN $EFRMCI IS SET TO ZERO.

$INSTR(@FARG,@CARG-1,@CARG-2,...)

@CARG-1 IS SEARCHED FOR 'OCCURRANCES' OF @CARG-2,

@CARG-3,... AND A VALUE OF TRUE IS RETURNED IF @CARG-2, @CARG-3,... ALL 'OCCUR' IN @CARG-1. OTHERWISE, A VALUE OF FALSE IS RETURNED. @CARG-N IS SAID TO 'OCCUR' IN @CARG-1 IF AT LEAST (&FLEN(@CARG-N)*@FARG)/100 CHARACTERS OF @CARG-N OCCUR CONTIGUOUSLY ANYWHERE IN @CARG-1.

$INSTRO(@FARG1,@CARG-1,@CARG-2,...)
THIS FUNCTION BEHAVES LIKE $INSTR WITH THE EXCEPTION THAT ANY 'OCCURANCE' OF @CARG-N MUST FOLLOW THE 'OCCURANCE' OF @CARG-(N-1) IN @CARG-1.

$CSBSTR(@CEXP,@FEXP-1,@FEXP-2)
A STRING OF CHARACTERS FROM @CEXP, BEGINNING WITH THE '@FEXP-1'TH AND CONTINUING FOR @FEXP-2 CHARCTERS IS RETURNED. IF @FEXP-1+@FEXP-2 EXCEEDS $FLEN(@CEXP) THEN ONLY THOSE CHARACTERS ACTUALLY IN @CEXP ARE RETURNED. IF @FEXP IS ZERO OR NEGATIVE, OR @FEXP-2 IS NEGATIVE THEN THE EXECUTION VARIABLE $FSBSTRI IS SET TO -1.

AND A NULL STRING IS RETURNED. IF @FEXP-2 IS ZERO THEN A NULL STRING IS RETURNED AND $FSBSTRI IS SET TO ZERO.

$CNXTWD(@CEXP-1,@FEXP<,@FVAR<,@CEXP-2>>)
THE NEXT 'WORD' IN @CEXP-1 IS RETURNED. @CEXP-1 IS SCANNED BEGINNING WITH THE CHARACTER AT POSITION @FEXP UNTIL THE FIRST CHARACTER WHICH IS *NOT* A DELIMITER IS FOUND. SUBSEQUENT CHARACTERS FORM THE RESULT UNTIL THE NEXT DELIMITER IS FOUND, AT WHICH TIME @FVAR IS SET (IF SPECIFIED) TO THE POSITION IN @CEXP-1 OF THE CHARACTER WHICH CAUSED THE SCAN TO END. IF THE END OF @CEXP-1 IS ENCOUNTERED DURING THE SCAN, THEN THE CHARACTERS PRIOR TO THE END OF THE STRING ARE RETURNED

AND @FVAR IS SET TO ZERO IF IT WAS SPECIFIED.
IF @CEXP-2 IS SPECIFIED THEN EACH CHARACTER IN @CEXP-2 IS USED AS A DELIMITER; OTHERWISE, THE FOLLOWING CHARACTERS CONSTITUTE THE 'DEFAULT' DELIMITERS.

BLANKS	.	!
?	=	((EXCLAMATION MARK)
+	,	-
)	*	/